



М.Н. Саушкин, А.Д. Зубкова

**О внедрении метаданных в
PDF-файлы на основе XMP-схемы
PRISM**

Рекомендуемая форма библиографической ссылки

Саушкин М.Н., Зубкова А.Д. О внедрении метаданных в PDF-файлы на основе XMP-схемы PRISM // Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23-28 сентября 2019 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2019. — С. 600-620. — URL: <http://keldysh.ru/abrau/2019/theses/79.pdf> doi:[10.20948/abrau-2019-79](https://doi.org/10.20948/abrau-2019-79)

Размещена также [презентация к докладу](#)

О внедрении метаданных в PDF-файлы на основе XMP-схемы PRISM

М.Н. Саушкин¹, А.Д. Зубкова¹

¹ Самарский государственный технический университет

Аннотация. Описывается разработка программного обеспечения metimpdf ([met]adata [im]port into a [pdf] file) для импорта метаданных в pdf-файлы с помощью XMP-схем. Для разработки программного обеспечения использовались среда разработки SharpDevelop, язык C# и библиотека iText. Программное обеспечение metimpdf работает в консольном режиме по аналогии с утилитой pdftk, которая в настоящее время используется некоторыми агрегаторами для внедрения метаинформации в pdf-файлы, например Math-Net.Ru, поэтому оно может использоваться на любом терминальном сервере для внедрения метаданных в pdf-файлы при их запросе с сервера. В отличие от утилиты pdftk, которая поддерживает только дублинское ядро (Dublin Core), metimpdf поддерживает XMP-схему Prism, которая может содержать полное библиографическое описание статьи, включая аннотацию и другие метаданные, по которым может производиться поисковый запрос.

Ключевые слова: метаданные, pdf-файл, xmp-схема Prism, metimpdf, библиотека iText, язык программирования C#

On the encapsulation of metadata in pdf files based on the Prism XMP Schema

M.N. Saushkin¹, A.D. Zubkova¹

¹ Samara State Technical University

Abstract. In this article, we describe the development of software for importing metadata into pdf files using XMP Schemas. We called this software metimpdf ([met]adata [im]port into a [pdf] file) utility. We programmed in C# language with SharpDevelop IDE, and iText library. The metimpdf utility is executed in console mode by analogy with the pdftk utility. Therefore, it can be used on any terminal server to import metadata into pdf files, when they are requested from the server. Unlike the pdftk utility, which supports only the Dublin Core, the metimpdf utility supports the Prism XMP Schema, which can contain a full bibliographic description of the article, including annotation and other metadata, which can be used for a search query.

Keywords: metadata, pdf file, Prism XMP Schema, metimpdf utility, iText library, C# language

1. Введение. Общая постановка задачи

Portable Document Format (pdf) имеет широкие возможности по созданию технической документации, электронных документов и является форматом де-факто для передачи научной информации. Международный стандарт ISO 15489-1:2016 [1] обязывает любой документ, кроме своего основного содержания, включать в себя метаданные, либо иметь связь с ними для необходимых действий с документом. Это необходимо для того, чтобы обеспечивать неизменность структуры документа при работе с ним. Метаданные также делают понятным обстоятельства, при которых документ создавался, был получен и использован.

Метаданные помогают обеспечить быструю обработку данных во многих областях. Они могут использоваться при настройке и обслуживании баз данных, создании документов. Если на начальном этапе создания документа, он будет дополнен метаданными, то это, несомненно, обязательно окупится в долгосрочной перспективе, так как метаданные дают множество возможностей для автоматизации процессов (которые не могут быть реализованы без них).

Международный стандарт ISO 15836-1:2017 [2] устанавливает, что любой информационный ресурс, в том числе и pdf-файл, должен иметь универсальный (минимальный) набор метаданных Dublin Core.

Особенно актуально использование метаданных в научных публикациях, так как количество публикаций растет так быстро, что ученый физически не успевает обратить внимание на новые исследования. Тысячи научных статей по разным направлениям публикуются еженедельно. Даже если интересы исследователя ограничены несколькими темами — практически невозможно охватить весь объем появляющейся информации.

Чрезмерная «перенасыщенность» информацией вызывает ситуацию, когда ученые инстинктивно сохраняют большинство pdf-файлов на «рабочий стол» компьютера, затем откладывают их в папку и подпапку для того, чтобы прочитать позже. Но большая часть этих файлов так и не бывает никогда прочитана, в основном из-за того, что поисковые запросы не позволяют быстро отыскать необходимую статью. Использование метаинформации в pdf-файлах позволяет частично решать данную проблему.

Сервисы-агрегаторы, которые работают с научными публикациями в формате pdf, например, Mendeley, ResearchGate, GoogleScholar и многие др., могут считывать метаданные о статье прямо из pdf-файлов (зачастую только англоязычных публикаций), минуя тем самым достаточно сложные процедуры анализа и разбора текста для выявления метаданных из текстового слоя pdf-файла статьи.

Практически все зарубежные издательства производят pdf-файлы с корректно заполненными метаданными. В отличие от них, в настоящее время

почти все российские издательства производят pdf-файлы, метаданные которых зачастую содержат лишь «мусор». Такие pdf-файлы поставляются сервисам-агрегаторам (научным библиотекам), таким как eLibrary.RU, CyberLeninka, и размещаются на сайтах издательств, откуда они в том же виде (без изменения метаданных) поступают к читателю.

Стороннее программное обеспечение, например утилита `pdftk` [3], позволяет внедрять метаинформацию в уже существующие pdf-файлы. Утилита `pdftk` успешно используется Общероссийским математическим порталом Math-Net.Ru для внедрения метаданных в pdf-файлы на основе имеющейся в базе данных портала информации: авторы статьи, название статьи, ключевые слова. В итоге, читатель получает от портала pdf-файлы с корректно заполненными метаданными.

Основной недостаток утилиты `pdftk` заключается в том, что она работает только с базовой (стандартной Dublin Core) XMP-схемой и не поддерживает другие, специализированные и более современные, которые могут содержать больше метаданных.

В настоящей работе описывается разработка программного обеспечения для интеграции метаданных в pdf-файлы с помощью XMP-схем Dublin Core и Prism [4], которая может содержать более полное библиографическое описание статьи, включая аннотацию, информацию о лицензировании и других атрибутах, которые могут быть использованы для описания научной публикации.

2. Используемое программное обеспечение

Для внедрения метаданных в pdf-файлы было решено использовать библиотеку `iText v. 5.5.13`, так как она обеспечивает поддержку самых передовых технологий в формате pdf. В том числе она поддерживает работу с любыми XMP-схемам, в том числе с необходимыми нам Dublin Core и Prism для внедрения соответствующих метаданных. Кроме этого, она наряду с коммерческой лицензией, распространяется и под лицензией Affero General Public License, а также обладает кросс-платформенностью. Именно лицензия AGPL и кросс-платформенность оказали решающую роль в выборе этой библиотеки для реализации поставленной задачи.

Библиотека `iText` изначально создавалась под язык программирования Java и была написана на этом же языке. Позже, из-за популярности библиотеки, она была портирована и на другие языки программирования, в том числе и на язык C#.

Исторически сложилось так, что полная документация и большинство примеров использования этой библиотеки встречается в основном на языке Java. Однако использование программ, написанных на языке Java, требует наличие виртуальной Java-машины, что может накладывать существенные ограничения на применение разрабатываемого программного обеспечения. Поэтому для написания кода программы `metimpdf ([met]adata [im]port into a`

[pdf] file) было решено воспользоваться библиотекой iText, портированной на язык C#.

Объектно-ориентированный язык программирования C# в настоящее время используется для разработки различных приложений для платформы Microsoft.NET Framework. Однако благодаря платформе Mono, которая поддерживается многими семействами операционных систем (Windows, Linux, BSD (FreeBSD, OpenBSD, NetBSD), Solaris, Mac OS X, Apple iOS, Wii), язык C# может быть использован для написания кроссплатформенных приложений.

После компиляции программного кода, написанного на языке программирования C#, по умолчанию получается приложение, которое имеет структуру исполняемых exe-файлов операционной системы Windows. Однако утилиты платформы Mono/GTK позволяют достаточно просто получить нативный исполняемый код в любой операционной системе. Например, в операционной системе Linux достаточно выполнить лишь одну команду

```
mkbundle -o hello hello.exe,
```

чтобы из исполняемого Windows-приложения hello.exe получить нативный исполняемый Linux-код hello.

В качестве основного инструмента для разработки была выбрана среда SharpDevelop, которая поддерживает разработку программного обеспечения на многих языках программирования, и была задумана как бесплатная и облегченная альтернатива для Microsoft Visual Studio (Visual Studio .NET). При этом выбранная среда разработки обладает почти всем набором функций, которые есть у Microsoft Visual Studio, и имеет форк на Mono/GTK — MonoDevelop, а сами проекты SharpDevelop и MonoDevelop полностью совместимы между собой, что гарантирует нам поддержку кроссплатформенности даже на уровне проектов (решений).

3. Реализация приложения metimpdf

В качестве схемы работы приложения metimpdf была выбрана схема, изображенная на рис. 1, которая аналогична схеме работы утилиты pdftk. Основное отличие заключается в том, что при работе утилиты pdftk не создаются промежуточные файлы. В нашем случае будет создаваться XMP-файл (который будет удаляться после завершения работы программы) исходя из логики, которая будет описана ниже.

Согласно предложенной схеме входными параметрами для программы являются два файла: pdf-файл, который не содержит метаданные или у которого в метаданных находится «мусор», а также текстовый файл с метаданными, имеющий определенную заранее оговоренную структуру.

Результатом работы приложения metimpdf будет pdf-файл с внедренными в него метаданными.

В соответствии с изложенным выше вызов (запуск) приложения будет осуществляться строкой

```
metimpdf source.pdf meta.data output.pdf,
```

где `source.pdf` — pdf-файл без метаданных, `meta.data` — текстовый файл с метаданными, `output.pdf` — выходной pdf-файл с метаданными.

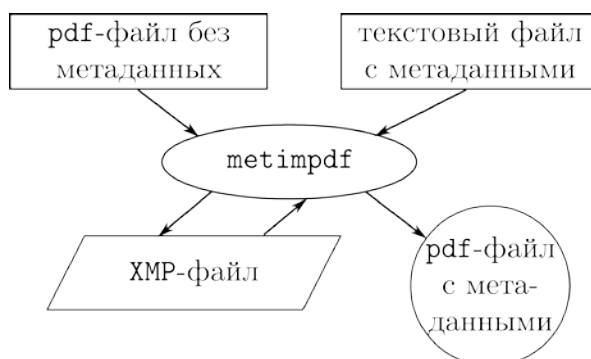


Рис. 1. Схема работы приложения `metimpdf`

В настоящем проекте метаданные берутся из обычного текстового файла, который может быть подготовлен вручную или с помощью программных средств, например, с помощью запросов из баз данных, в которых эти метаданные уже присутствуют.

Структура файла простая и может быть описана следующим образом.

- Каждый элемент данных записывается в одной строке (разрыв строк не допускается).
- Элемент начинается с его названия, например, `Title`, `Author`, `Doi` и др., которые соответствуют идентификаторам в XMP-схемах, после которого стоит знак `=` и в фигурных скобках идет значение, которое должен принять соответствующий идентификатор XMP-схемы. Например, `Title = {О скорости стабилизации решений задачи Коши}`.
- В случае, если элементы добавляются в XMP-схему в виде коллекций, то они разделяются точкой с запятой, окруженной пробелами. К таким данным относятся элементы с идентификаторами `Author` и `Keywords`.

Для описания метаданных могут быть использованы следующие элементы, представленные в таблице, которые в текстовом файле `meta.data` могут идти в различном порядке. Следует отметить, что некоторые элементы используются в обеих XMP-схемах — `Dublin Core` и `Prism`.

Элемент	Присутствие в XMP-схемах	Описание элемента
<code>Title</code>	<code>Dublin Core</code> , <code>Prism</code>	Название статьи (документа)
<code>Author</code>	<code>Dublin Core</code> , <code>Prism</code>	Автор(ы) статьи (документа); может добавляться XMP-схему в виде коллекции
<code>Keywords</code>	<code>Dublin Core</code> , <code>Prism</code> , <code>pdf</code>	Ключевые слова статьи (документа); добавляется XMP-схему в виде коллекции
<code>Doi</code>	<code>Prism</code>	Идентификатор DOI статьи (документа)
<code>DoiUrl</code>	<code>Prism</code>	Стандартная ссылка (URL) для доступа к статье

		(документу) по идентификатору DOI
Publisher	Dublin Core	Издатель статьи (документа)
Rights	Dublin Core, xmpRights	Текстовое поле с описанием исключительных прав, прав доступа и использования (распространения) статьи (документа)
RightsUrl	xmpRights	Ссылка (URL) на web-документ, в котором приводится описание прав доступа и использования (распространения) статьи (документа)
Description	Dublin Core, Prism	Описание (аннотация) статьи (документа)
Issn	Prism	Номера ISSN издания, в котором опубликована статья (документ)
Volume	Prism	Том издания, в котором опубликована статья (документ)
Number	Prism	Номер издания, в котором опубликована статья (документ)
CoverDisplay Date	Prism	Текстовый формат описания даты выхода издания, в котором опубликована статья (документ), например, Март, 2017
CoverDate	Prism	Машиночитаемый формат описания даты выхода издания, в котором опубликована статья (документ), например, 2017-06-22
IssueName	Prism	Название издания, в котором опубликована статья (документ)
PageRange	Prism	интервал страниц на которых опубликована статья (документ) в издании
StartingPage	Prism	Первая страница из интервала страниц на которых опубликована статья (документ) в издании
EndingPage	Prism	Последняя страница из интервала страниц на которых опубликована статья (документ) в издании
CreatorTool	xmp	Инструментарий с помощью которого сформирован (создан) pdf-файл с метаданными
Producer	xmp	Распространитель (создатель) метаданных pdfфайла

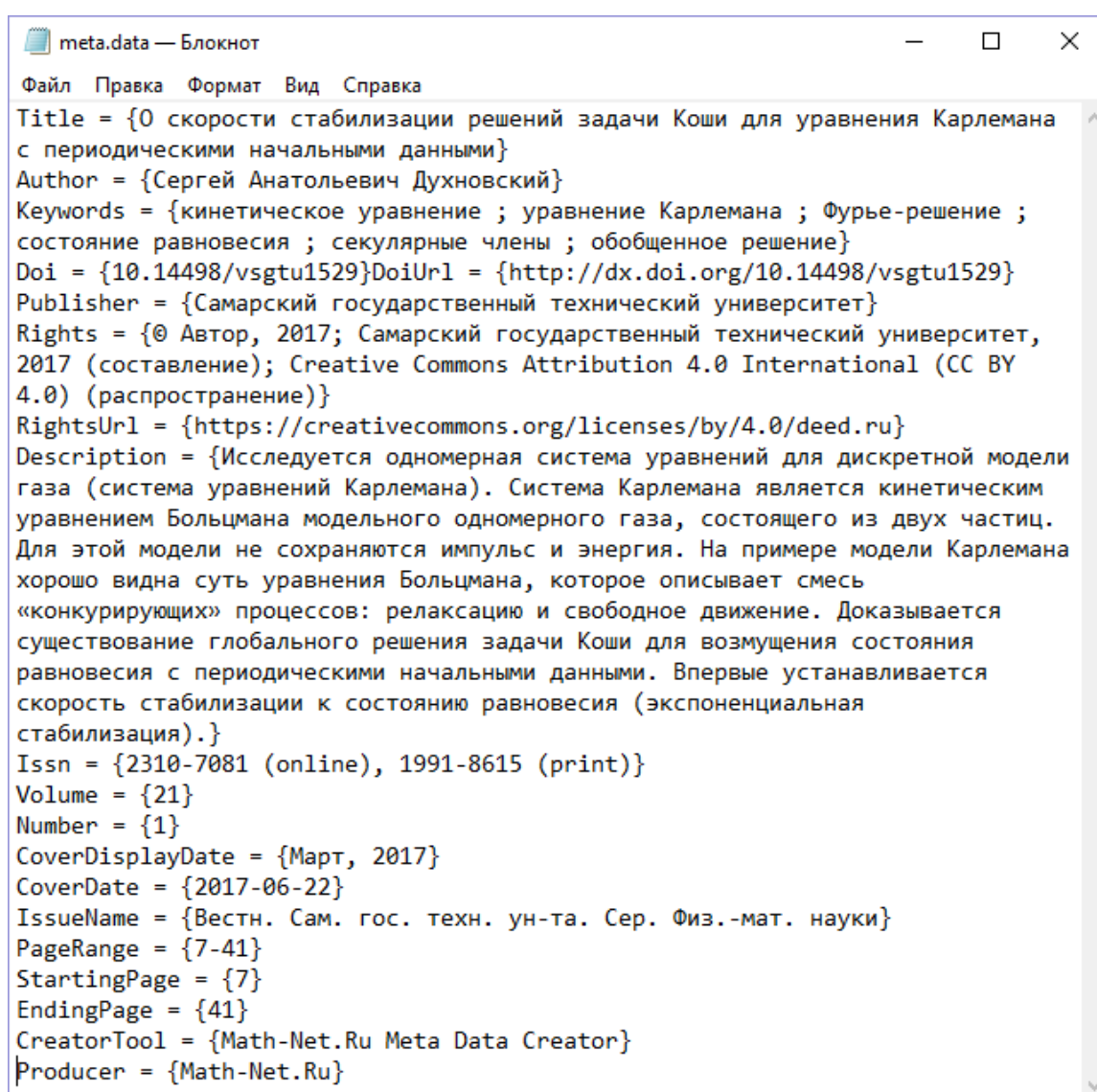
Пример файла с метаданными, который использовался для тестового прогона программы metimpdf, приведен на рис. 2.

В соответствии с логикой приложения, изложенной выше, из текстового файла с метаданными необходимо прочитать элементы XMP-схемы и их значения, другими словами создать карту из элементов и их значений. Для удобства работы (чтения, записи и хранения карты) сразу был создан вспомогательный класс `MapObject` (см. листинге на рис. 3). Перед использованием класса `MapObject` должна быть создана и зарегистрирована

схема Prism, так как она не существует в пространстве имен iTextSharp.xmp библиотеки iText (см. рис. 4).

После регистрации схемы Prism можно создать карту (класс map) для работы с элементами текстового файла с метаданными. Создание карты map осуществляется с помощью кода, представленного в листинге на рис. 5. Имена объектов карты соответствуют именам элементов из таблицы, приведенной выше.

Чтение данных из текстового файла с метаданными осуществляется построчно с использованием метода ReadAllLines класса File. После этого осуществляется перебор и разбор прочитанных строк для создания словаря «ключ : значение» (info — экземпляр класса Dictionary). Код чтения данных из текстового файла и создания словаря info приведен в листинге на рис. 6.



```
meta.data — Блокнот
Файл  Правка  Формат  Вид  Справка
Title = {0 скорости стабилизации решений задачи Коши для уравнения Карлемана
с периодическими начальными данными}
Author = {Сергей Анатольевич Духновский}
Keywords = {кинетическое уравнение ; уравнение Карлемана ; Фурье-решение ;
состояние равновесия ; секулярные члены ; обобщенное решение}
Doi = {10.14498/vsgtu1529}DoiUrl = {http://dx.doi.org/10.14498/vsgtu1529}
Publisher = {Самарский государственный технический университет}
Rights = {© Автор, 2017; Самарский государственный технический университет,
2017 (составление); Creative Commons Attribution 4.0 International (CC BY
4.0) (распространение)}
RightsUrl = {https://creativecommons.org/licenses/by/4.0/deed.ru}
Description = {Исследуется одномерная система уравнений для дискретной модели
газа (система уравнений Карлемана). Система Карлемана является кинетическим
уравнением Больцмана модельного одномерного газа, состоящего из двух частиц.
Для этой модели не сохраняются импульс и энергия. На примере модели Карлемана
хорошо видна суть уравнения Больцмана, которое описывает смесь
«конкурирующих» процессов: релаксацию и свободное движение. Доказывается
существование глобального решения задачи Коши для возмущения состояния
равновесия с периодическими начальными данными. Впервые устанавливается
скорость стабилизации к состоянию равновесия (экспоненциальная
стабилизация).}
Issn = {2310-7081 (online), 1991-8615 (print)}
Volume = {21}
Number = {1}
CoverDisplayDate = {Март, 2017}
CoverDate = {2017-06-22}
IssueName = {Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки}
PageRange = {7-41}
StartingPage = {7}
EndingPage = {41}
CreatorTool = {Math-Net.Ru Meta Data Creator}
Producer = {Math-Net.Ru}
```

Рис. 2. Пример файла с метаданными, который использовался для тестового прогона программы metimpdf


```

class MapObject{
    // конструктор класса без параметров
    public MapObject(){ }
    // конструктор класса без параметра сокращенного имени(для инициализации
в случае если другое имя такое же как и оригинальное)
    public MapObject(string schemaNs, string name, ValueType valueType, bool
isSingle = true) : this(schemaNs, name, name, valueType, isSingle){ }
    /**
    * конструктор со всеми параметрами
    * schemaNs - схема принадлежности свойства (prism, dc, xmp, xmpRights и
т.д. все они (кроме prism) есть в объекте констант XmpConst)
    * name - полное имя свойства (совпадает с именем в data файле)
    * valueName - сконвертированное имя (на случай если имя в выходном
файле отличается от оригинального имени)
    * valueType - тип свойства (Объект перечисление ValueType, один из трёх
типов массива или просто однострочное свойство)
    * isSingle - работает только если объект массив элементов и отвечает за то,
следует ли распилить элемент и вставить его коллекцией или одной строкой
    */
    public MapObject(string schemaNs, string name, string valueName, ValueType
valueType, bool isSingle = true){
        XmlSchemaNs = schemaNs;
        Name = name;
        ValueName = valueName;
        ValueType = valueType;
        IsSingle = isSingle;
    }

    public string XmlSchemaNs { get; set; }
    public string Name { get; set; }
    public string ValueName { get; set; }
    public ValueType ValueType { get; set; }

    public bool IsSingle { get; set; }
}

```

Рис. 3. Реализация вспомогательного класса MapObject

```

string prismSchema = "http://prismstandard.org/namespaces/basic/2.2/";
XmpMetaFactory.SchemaRegistry.RegisterNamespace(prismSchema, "prism");

```

Рис. 4. Регистрация схемы Prism

```

List<MapObject> map = new List<MapObject>();
map.Add(new MapObject(XmpConst.NS_DC, "Title", "title", ValueType.Alt));
map.Add(new MapObject(XmpConst.NS_DC, "Author", "creator", ValueType.Seq,
false));
map.Add(new MapObject(XmpConst.NS_DC, "Keywords", "subject", ValueType.Bag,
false));
map.Add(new MapObject(XmpConst.NS_DC, "Doi", "identifier", ValueType.Property));
map.Add(new MapObject(XmpConst.NS_DC, "Publisher", "publisher", ValueType.Bag));
map.Add(new MapObject(XmpConst.NS_DC, "Rights", "rights", ValueType.Alt));
map.Add(new MapObject(XmpConst.NS_DC, "Description", "description",
ValueType.Alt));

map.Add(new MapObject(XmpConst.NS_PDF, "Keywords", ValueType.Property));
map.Add(new MapObject(XmpConst.NS_PDF, "Producer", ValueType.Property));

map.Add(new MapObject(XmpConst.NS_XMP, "CreatorTool", ValueType.Property));

map.Add(new MapObject(XmpConst.NS_XMP_RIGHTS, "Rights", "UsageTerms",
ValueType.Alt));
map.Add(new MapObject(XmpConst.NS_XMP_RIGHTS, "RightsUrl", "WebStatement",
ValueType.Property));

map.Add(new MapObject(prismSchema, "Doi", "doi", ValueType.Property));
map.Add(new MapObject(prismSchema, "DoiUrl", "url", ValueType.Property));
map.Add(new MapObject(prismSchema, "Issn", "issn", ValueType.Property));
map.Add(new MapObject(prismSchema, "Volume", "volume", ValueType.Property));
map.Add(new MapObject(prismSchema, "Number", "number", ValueType.Property));
map.Add(new MapObject(prismSchema, "CoverDisplayDate", "coverDisplayDate",
ValueType.Property));
map.Add(new MapObject(prismSchema, "CoverDate", "coverDate",
ValueType.Property));
map.Add(new MapObject(prismSchema, "IssueName", "issueName",
ValueType.Property));
map.Add(new MapObject(prismSchema, "PageRange", "pageRange",
ValueType.Property));
map.Add(new MapObject(prismSchema, "StartingPage", "startingPage",
ValueType.Property));
map.Add(new MapObject(prismSchema, "EndingPage", "endingPage",
ValueType.Property));

```

Рис. 5. Создание карты элементов

После чтения данных из текстового файла с метаданными и создания словаря `info` («ключ : значение»), согласно логике приложения, изложенной выше, необходимо перейти к созданию XMP-файла, через который метаданные будут записываться в pdf-файл. Создание XMP-файла осуществляется методами класса `XmpWriter` и представлено в листинге на рис. 7.

Здесь (в листинге на рис. 7) `fs` — экземпляр класса `FileStream` (файлового потока), который связан с физическим файлом `metadata.xmp`. Этот файл,

согласно схеме (см. рис. 1), является промежуточным и содержит XMP-схему с метаданными.

Для записи значений карты в узлы дерева XMP используется метод `SetValue`. Реализация этого метода из-за его громоздкости здесь не приводится. В этом методе реализован механизм правильной записи значений элементов (`Author`, `Keywords`), которые являются коллекциями (их значения состоят из нескольких значений).

Внедрение XMP-схемы из промежуточного файла `metadata.xmp` в исходный pdf-файл `SourcePdfFile` и запись результата в pdf-файл `OutputPdfFile` осуществляется методами классов `PdfReader`, `PdfStamper` библиотеки `iText` и не представляет особой сложности (см. листинг на рис. 8).

```
string[] lines = File.ReadAllLines(MetadataFile);

// Создаем словарь разбора строк в сопоставлении ключ:значение
Dictionary<string, string> info = new Dictionary<string, string>();

// цикл перебора строк для разбора в словарь ключ:значение
foreach(string line in lines){
string[] splitted = line.Split('='); // делим строку пополам относительно знака =
string key = splitted[0].Trim(); // ключ(значение слева от = )
string value = splitted[1].Trim(); // значение(значение справа от =)
value = value.Substring(1, value.Length - 2); // убираем скобки {}
info.Add(key, value);
}
```

Рис. 6. Чтение данных из текстового файла и создание словаря `info`

4. Демонстрация работы программы `metimpdf` и иллюстрация результатов работы

В соответствии с логикой приложения, программа `metimpdf` является консольным приложением, которая выполняется при корректном задании параметров. Запуск приложения осуществляется строкой

```
metimpdf source.pdf meta.data output.pdf,
```

где `source.pdf` — pdf-файл без метаданных, `meta.data` — текстовый файл с метаданными, `output.pdf` — выходной pdf-файл с метаданными.

Проиллюстрируем работу приложения `metimdf`. Для этого возьмем pdf-файл в котором отсутствуют метаданные и инкапсулируем (импортируем) в него подготовленные метаданные (см. рис. 2). Отметим, что подготовка файла с метаданными не входила в задачу данного проекта. Этот файл может быть подготовлен любыми средствами, в том числе с помощью программных средств портала `Math-Met.Ru`.

```

XmpWriter xmpWriter = new XmpWriter(fs);

// цикл перебора словаря для дальнейшего разбора
foreach (string key in info.Keys){
    // цикл перебора карты на поиск соответствующих ключевым словам
    объектов
    foreach(MapObject p in map){
        if(p.Name == key){ // если имя объекта равно ключевому слову из data,
        то
            /*
            * сохраняем значение в xmpWriter с помощью
            специального
            * упрощенного метода сохранения (сам метод
            создан в конце класса)
            */
            SetValue(xmpWriter, p, info[key]);
        }
    }
}
// модификатор языка
XmpNode languageQualifier = new XmpNode("xml:lang", "x-default", new
PropertyOptions {Qualifier = true });

// более низкий уровень взаимодействия с деревом xml для проставки
дополнительных атрибутов
XmpNode groupNode = (xmpWriter.XmpMeta as
XmpMetaImpl).Root.FindChildByName(XmpConst.NS_DC);

// находим элементы которым нужно указать языковую конструкцию
XmpNode titleDcNode = groupNode.FindChildByName("dc:title");
XmpNode rightsDcNode = groupNode.FindChildByName("dc:rights");
XmpNode descriptyopmDcNode = groupNode.FindChildByName("dc:description");

// проставляем модификатор
titleDcNode.GetChild(1).AddQualifier(languageQualifier);
rightsDcNode.GetChild(1).AddQualifier(languageQualifier);
descriptyopmDcNode.GetChild(1).AddQualifier(languageQualifier);

groupNode = (xmpWriter.XmpMeta as
XmpMetaImpl).Root.FindChildByName(XmpConst.NS_XMP_RIGHTS);
XmpNode rightsXmpRightsNode =
groupNode.FindChildByName("xmpRights:UsageTerms");
rightsXmpRightsNode.GetChild(1).AddQualifier(languageQualifier);

// Закрываем xmpWriter
xmpWriter.Close();

```

Рис. 7. Создание XMP-файла

```

PdfReader reader = new PdfReader(SourcePdfFile);
PdfStamper stamper = new PdfStamper(reader, new FileStream(OutputPdfFile,
FileMode.Create));
using(FileStream fs = File.OpenRead("metadata.xmp")){
    byte[] buffer = new byte[fs.Length];
    fs.Read(buffer, 0, buffer.Length);
    stamper.XmpMetadata = buffer;
}

    stamper.Close();
    reader.Close();
}

```

Рис. 8. Внедрение XMP-схемы в pdf-файл

Большинство программ, которые просматривают pdf-файлы, могут выводить метаданные документа, соответствующие схеме Dublin Core. Поэтому для просмотра метаданных была использована ознакомительная версия программы Adobe Acrobat DC. На рис. 9 приведен скриншот окна программы Adobe Acrobat DC со свойствами документа без метаданных. В этом окне отображается метаинформация, которая соответствует XMP-схеме Dublin Core.

Для просмотра метаинформации, соответствующей другим XMP-схемам, необходимо перейти на вкладку «Дополнительные метаданные». На рис. 10, 11 приведены скриншоты со свойствами документа вкладки «Дополнительные метаданные». Эти скриншоты подтверждают, что исходный файл не содержит метаданных. В исходном pdf-файле присутствуют только данные из набора обязательных XMP-схем (xmp, pdf, pdfx). Часть этих метаданных никогда не изменяется, другая часть может модифицироваться.

На рис. 12–15 приведены скриншоты этих же окон для файла output.pdf, который был получен из исходного pdf-файла с помощью программы metimpdf. Отметим, что на рис. 14 отображаются данные, соответствующие XMP-схеме Dublin Core, а на рис. 15 — схеме Prism. Информация, представленная на этих скриншотах, полностью соответствует исходным данным из файла с метаданными (см. рис. 2.).

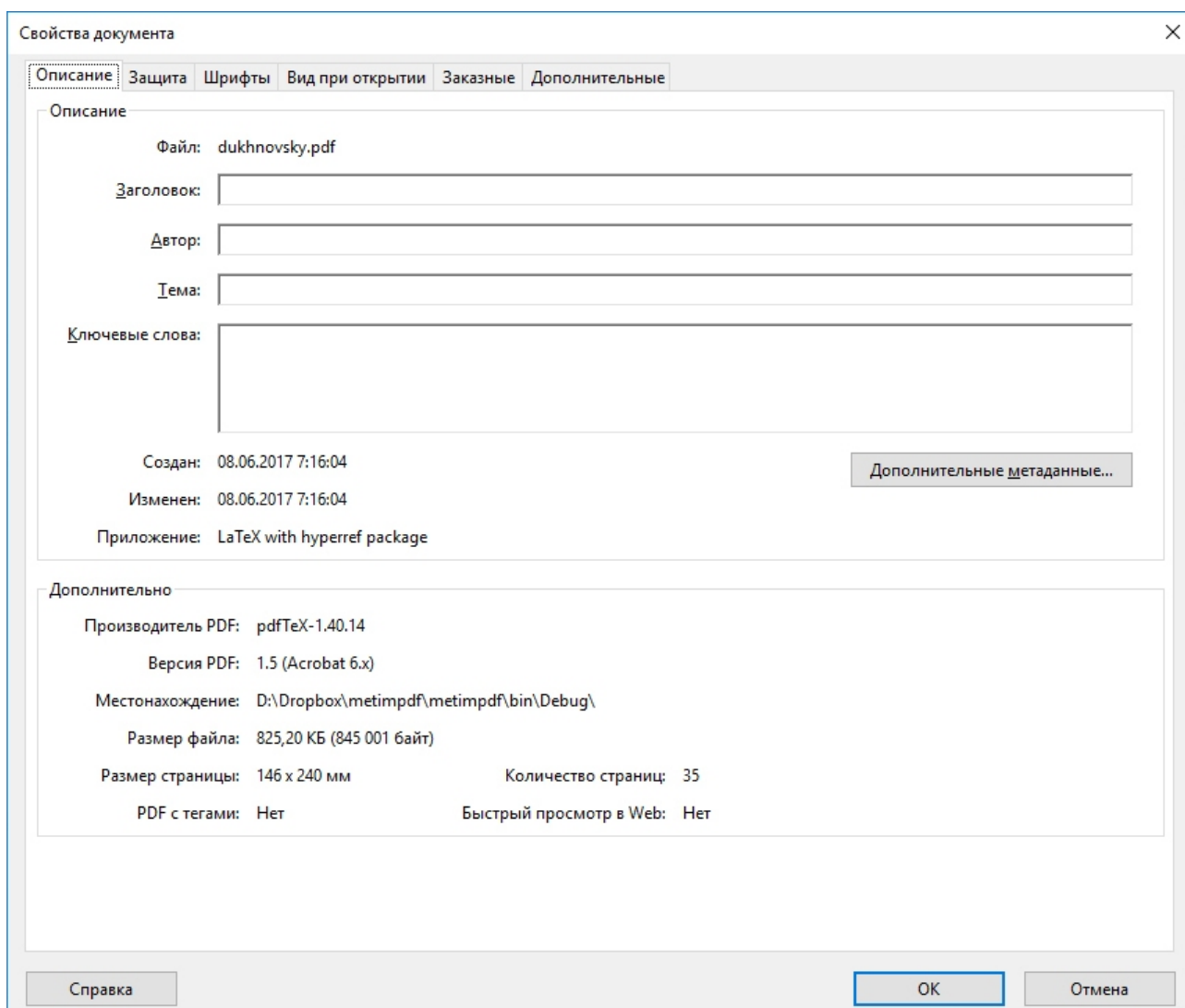


Рис. 9. Скриншот окна со свойствами документа без метаданных (Dublin Core)

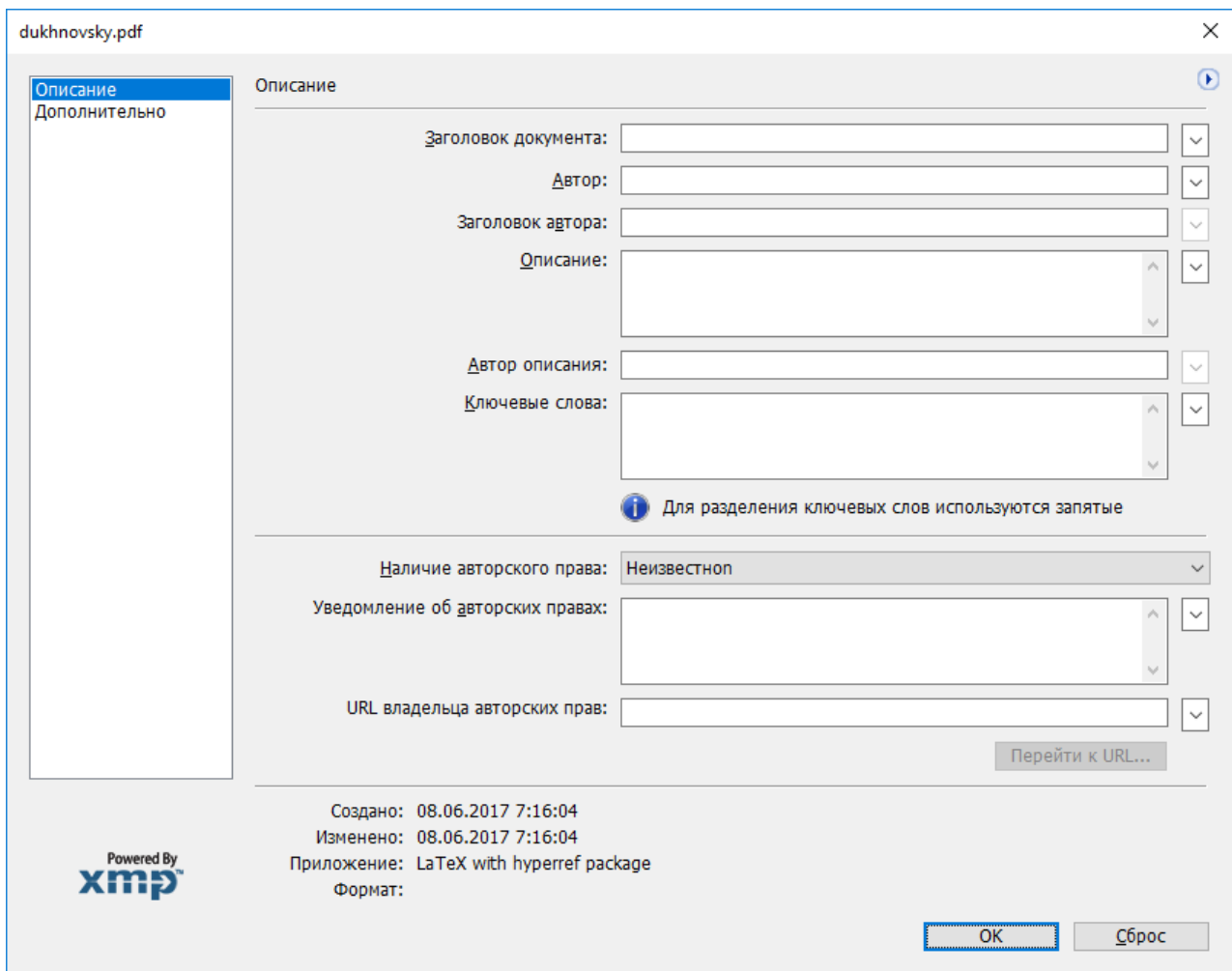


Рис. 10. Скриншот окна со свойствами документа без метаданных (Дополнительные метаданные – описание)

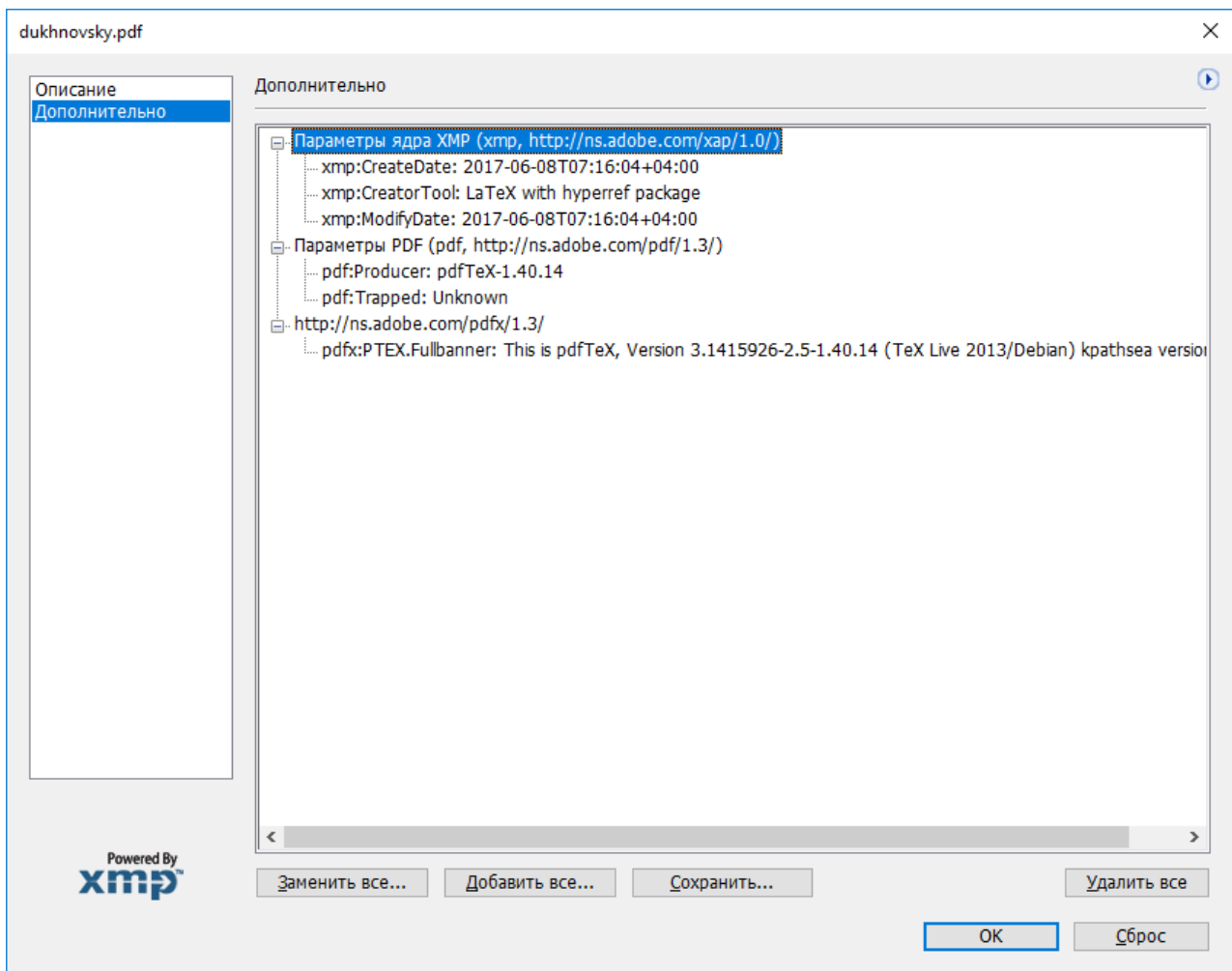


Рис. 11. Скриншот окна со свойствами документа без метаданных
(Дополнительные метаданные – дополнительно)

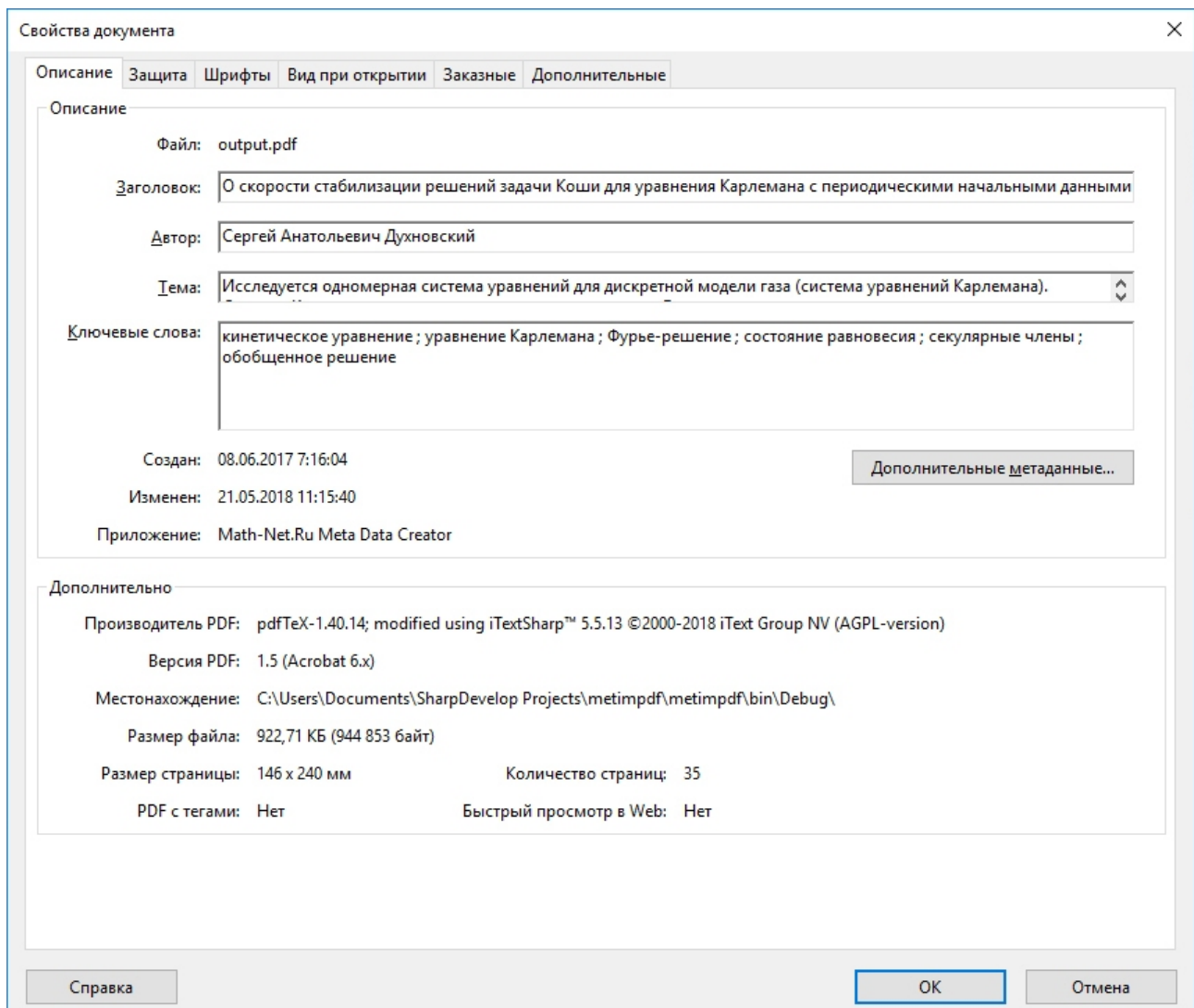


Рис. 12. Скриншот окна свойств документа с метаданными (Dublin Core) после обработки исходного файла программой metimpdf

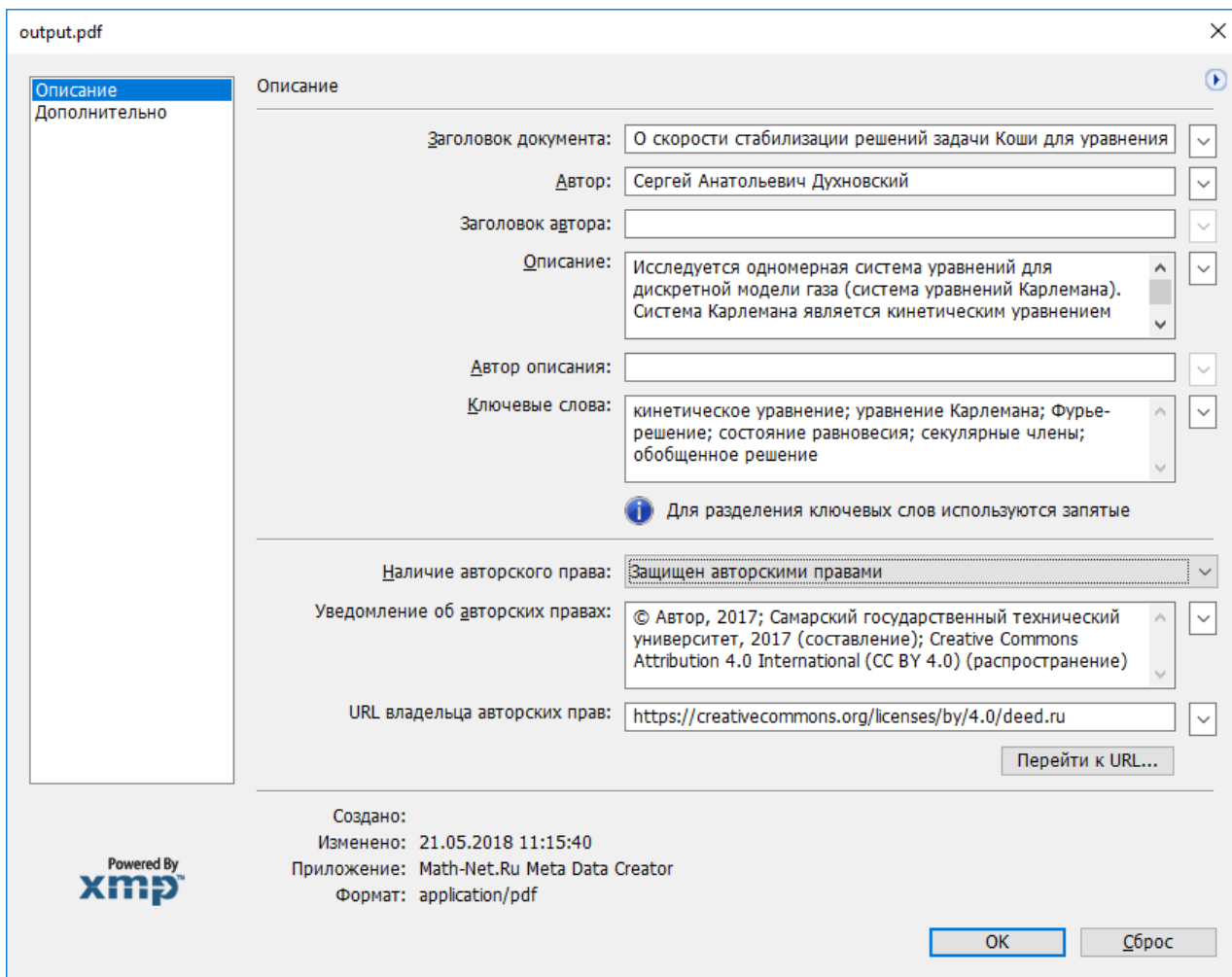


Рис. 13. Скриншот окна свойств документа с метаданными (Дополнительные метаданные – описание) после обработки исходного файла программой metimpdf

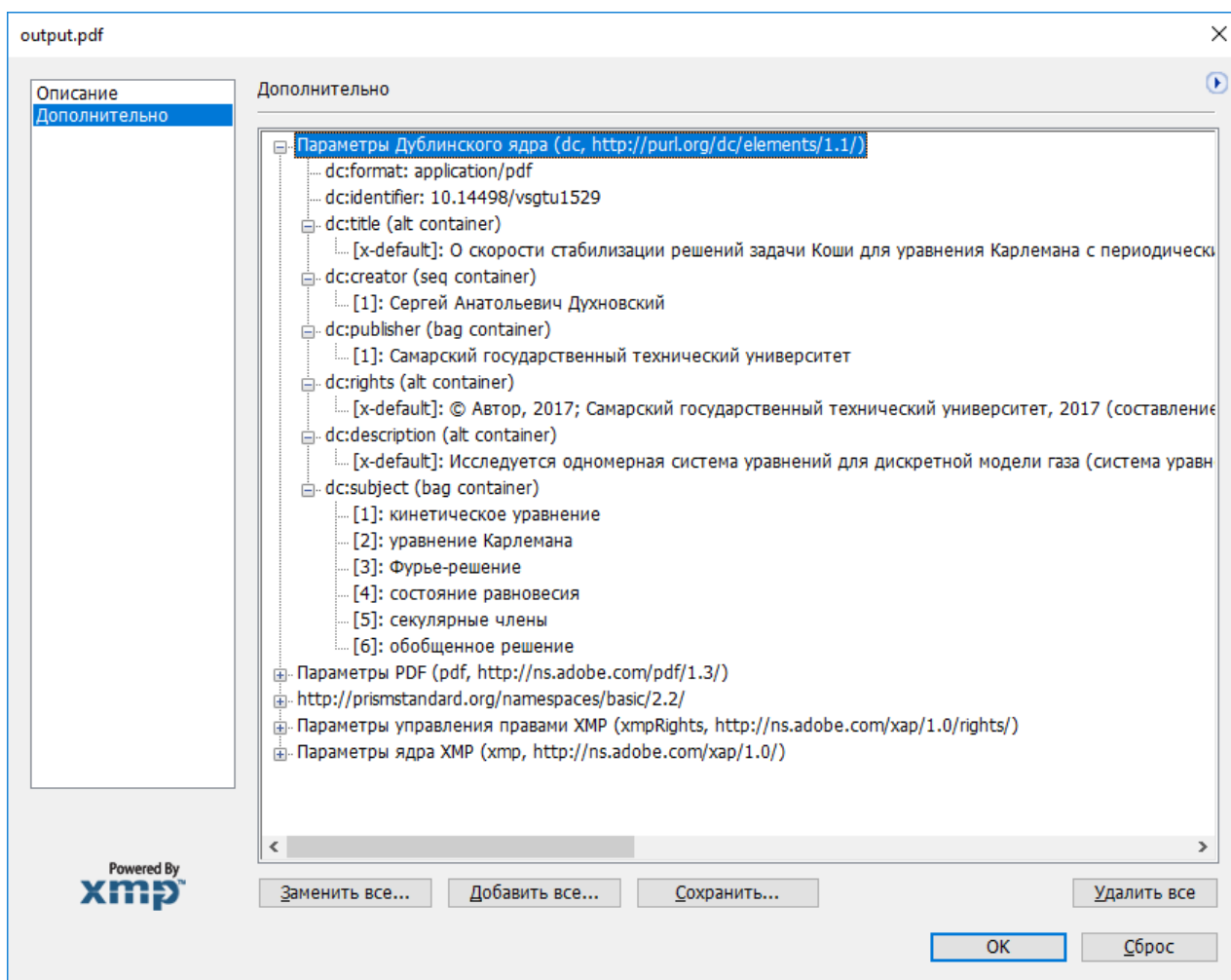


Рис. 14. Скриншот окна свойств документа с метаданными (Дополнительные метаданные – дополнительно) после обработки исходного файла программой metimpdf (XMP-схема Dublin Core)

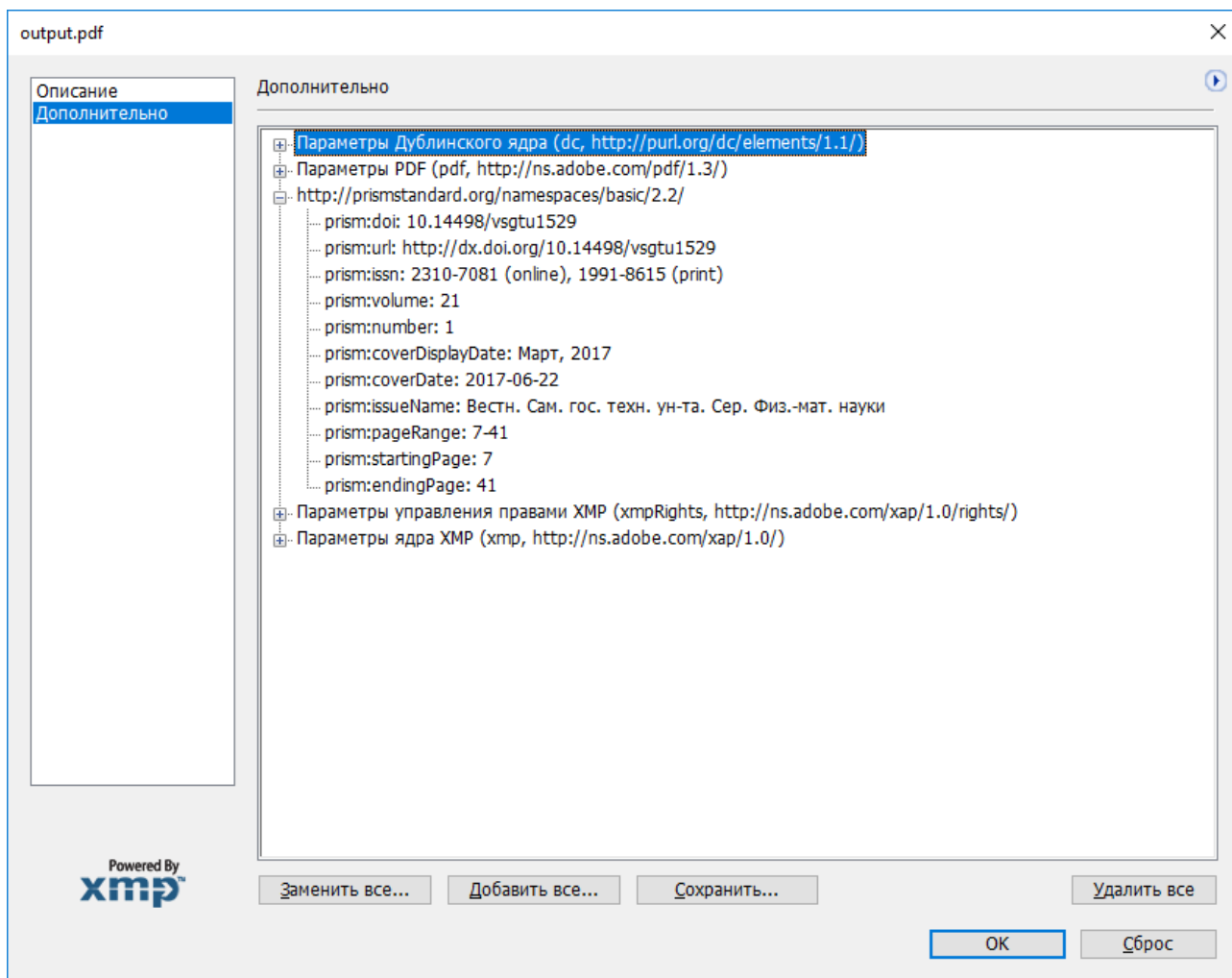


Рис. 15. Скриншот окна свойств документа с метаданными (Дополнительные метаданные – дополнительно) после обработки исходного файла программой `metimpdf` (XMP-схема Prism)

5. Заключение

В статье описана разработка программного обеспечения `metimpdf` (`[met]adata [im]port into a [pdf] file`) для импорта метаданных в pdf-файлы с помощью XMP-схем и продемонстрирована его работа.

Для разработки программного обеспечения использовались среда разработки `SharpDevelop v. 5.1`, язык `C#` и библиотека `iText v. 5.5.13`.

Программное обеспечение `metimpdf` выполняется в консольном режиме по аналогии с утилитой `pdftk`, которая в настоящее время используется некоторыми агрегаторами для внедрения метаинформации в pdf-файлы, например `Math-Net.Ru`, и поэтому оно может использоваться на любом терминальном сервере для внедрения метаданных в pdf-файлы при их запросе с сервера.

В отличие от утилиты `pdftk`, которая поддерживает только дублинское ядро (Dublin Core), `metimpdf` поддерживает XMP-схему Prism, которая может

содержать полное библиографическое описание статьи, включая аннотацию и другие метаданные, по которым может производиться поисковый запрос.

Корректное отображение метаданных в выходных файлах, их соответствие исходным данным из файла с метаданными, а также отсутствие ошибок и сообщений о нарушении целостности pdf-файла при его просмотре позволяет сделать вывод, что разработанный программный продукт *metimpdf* работает корректно и соответствует поставленным целям и задачам.

Работа выполнена в рамках договоров МОН 2018/20 от 20 апреля 2018 г. и МОН 2019/40 от 29 мая 2019 г. на реализацию программы развития научного журнала «Вестник Самарского государственного технического университета. Серия «Физико-математические науки» между ФГБОУ ВО «СамГТУ» и НП «НЭИКОН», действующего в рамках Государственного контракта на выполнение работ (оказание услуг) для государственных нужд от 28 августа 2017 г. № 14.597.11.0035 по проекту «Продолжение конкурсной поддержки программ развития научных журналов с целью их вхождения в международные наукометрические базы данных» в рамках реализации мероприятия 3.3.1 федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014–2020 годы», утвержденной постановлением Правительства Российской Федерации от 21 мая 2013 г. № 426 (Реестровый номер Государственного контракта 1771053913517000299).

Литература

1. ISO 15489-1:2016 Information and documentation – Records management – Part 1: Concepts and principles: —
URL: <https://www.iso.org/standard/62542.html>.
2. ISO 15836-1:2017 Information and documentation – The Dublin Core metadata element set – Part 1: Core elements. —
URL: <https://www.iso.org/standard/71339.html>.
3. PDFtk – The PDF Toolkit. — URL: <https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/>.
4. PRISM: Publishing Requirements for Industry Standard Metadata. —
URL: http://www.prismstandard.org/specifications/3.0/PRISM_Introduction_3.0.htm.

References

1. ISO 15489-1:2016 Information and documentation – Records management – Part 1: Concepts and principles: —
URL: <https://www.iso.org/standard/62542.html>.
2. ISO 15836-1:2017 Information and documentation – The Dublin Core metadata element set – Part 1: Core elements. —
URL: <https://www.iso.org/standard/71339.html>.

3. PDFtk – The PDF Toolkit. — URL: <https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/>.
4. PRISM: Publishing Requirements for Industry Standard Metadata. — URL: http://www.prismstandard.org/specifications/3.0/PRISM_Introduction_3.0.htm.